

# Laboratório 2 - Computação Concorrente

Ricardo Kaê - DRE 116 039 521

**Relatório do Laboratório 2** Para o Laboratório 2 foram feitos os seguintes arquivos:

- *lab2.c*  $\rightarrow$  *lab2* (Executável)
- *tabela.sh*  $\rightarrow$  *tabela.txt*, *dataset - m500.dat*, *dataset - m1000.dat*, *dataset - m2000.dat*  
Um script em bash para automatizar todos os casos de execução e gerar quatro arquivos de dados: Uma tabela e mais três conjuntos de dados para serem plotados.
- *gera\_graficos.gnu*  
Um script do *gnuplot* para gerar gráficos a partir dos arquivos *.dat*

## Código *lab2.c*

O código *lab2.c* multiplica duas matrizes quadradas  $(A, B)$  de ordem definida pelo usuário (*dim*), de maneira sequencial e de maneira concorrente, onde de maneira concorrente, o usuário decide o número de threads a ser usado pelo programa.

Ambas as matrizes  $(A, B)$  são inicializadas com valor  $i + j$  em todas as entradas, onde  $i, j \in [0, dim] \subset \mathbb{Z}$ .

O resultado da multiplicação é armazenado numa matriz  $S$  e  $C$ , quando feito respectivamente, sequencialmente e concorrentemente.

Além disso, o código conta com três funções: *verifica\_mult()*, *print\_matrix()*, *gera\_tabela()*.

A função *verifica\_mult()* verifica se as matrizes  $S, C$  após as multiplicações são iguais. Isto é, se o procedimento sequencial confere com procedimento concorrente.

A função *print\_matrix()* imprime o número de threads e as matrizes de entrada e saída. Só vale ser usada quando as matrizes tem ordem pequena. Para matrizes de ordem grande, a saída é ruim de ver.

E por fim, a função *gera\_tabela()* que confere ao programa a saída adequada para quaisquer ordens de matrizes. Portanto, a função *print\_matrix()* encontra-se comentada no código (linha 134).

A saída de *gera\_tabela()* imprime cinco valores tabulados, que são respectivamente:

#Tseq, #Thread Principal, #Tconc, #Nº de threads, #Ganho (1)

A impressão é feita sem rotular as colunas, então apenas os valores são impressos.

Essa saída é auxiliada pelo script *tabela.sh*, para gerar uma tabela de valores automaticamente para diferentes casos de execução do programa, isto é, matrizes com diferentes ordens e com diferentes números de threads.

### **Script** *tabela.sh*

O script *tabela.sh* cumpre dois objetivos:

1. Gerar automaticamente uma tabela com as colunas de (1) (resultado da função *gera\_tabela()* de *lab2.c*), para diferentes casos de execução. Isto é, 5 execuções para matrizes de ordem 500, 1000 e 2000. E para cada uma das execuções, registrar o tempo de computação com 1, 2 e 4 threads respectivamente. Assim, para cada thread, há 5 execuções, totalizando 15 execuções no total. Como há 3 matrizes, são 15 execuções para cada uma, totalizando 45 execuções concorrentes no total. E como para cada execução concorrente, o programa *lab2.c* faz uma multiplicação sequencial também, são 90 multiplicações de matrizes ao todo.

\* O script costuma demorar em torno de 20 a 40 min para entregar o resultado final, que é redirecionado para um arquivo de texto *tabela.txt*

2. Gerar automaticamente conjuntos de dados para serem plotados no *gnuplot*.

Três conjuntos de dados são gerados (*dataset - m500.dat*, *dataset - m1000.dat*, *dataset - m2000.dat*), cujas as informações são retiradas de *tabela.txt*. Um conjunto de dados é referente a matriz com dimensão 500, outro com a dimensão 1000 e o último com a dimensão 2000.

A partir desses arquivos, pode-se executar o script *gera-graficos.gnu* dentro do *gnuplot*, que ele gera como saída três arquivos *.eps*: *m500.eps*, *m1000.eps*, *m2000.eps*, que são os gráficos com os tempos de execução sequencial e concorrente das matrizes de dimensão 500, 1000 e 2000 respectivamente.

O arquivo *tabela.txt*

# TABELA #

# MATRIZ COM DIMENSÃO 500

# T Sequencial (em s)	#Thread Principal	#T Concorrente (em s)	#n° Threads	#Ganho
1.322899	1	1.468214	1	0.901026
1.311620	1	1.440662	1	0.910428
1.339205	1	1.436752	1	0.932105
1.290908	1	1.462699	1	0.882552
1.317734	1	1.418772	1	0.928785
1.285275	1	0.736922	2	1.744114
1.334348	1	0.839372	2	1.589698
1.369319	1	0.731159	2	1.872806
1.215496	1	0.746455	2	1.628358
1.210268	1	0.720318	2	1.680186
1.204502	1	0.408765	4	2.946689
1.266952	1	0.419967	4	3.016788
1.215388	1	0.410308	4	2.962135
1.208371	1	0.418479	4	2.887529
1.200675	1	0.408955	4	2.935961

# MATRIZ COM DIMENSÃO 1000

# T Sequencial (em s)	#Thread Principal	#T Concorrente (em s)	#n° Threads	#Ganho
12.852179	1	15.025679	1	0.855348
9.170262	1	9.515158	1	0.963753
8.925299	1	9.176126	1	0.972665
9.154169	1	9.483944	1	0.965228
9.257833	1	9.600310	1	0.964326
9.156375	1	5.344537	2	1.713221
12.842645	1	7.545107	2	1.702116
9.319945	1	5.340889	2	1.745018
9.220565	1	5.325122	2	1.731522
9.387186	1	5.386652	2	1.742675
9.323905	1	3.035821	4	3.071296
9.287894	1	3.046447	4	3.048762
9.252381	1	3.029967	4	3.053624
9.437469	1	3.089035	4	3.055152
9.254626	1	3.012638	4	3.071934

# MATRIZ COM DIMENSÃO 2000

# T Sequencial (em s)	#Thread Principal	#T Concorrente (em s)	#n° Threads	#Ganho
80.642155	1	84.815525	1	0.950795
79.440313	1	84.737322	1	0.937489
80.644480	1	84.234690	1	0.957378
80.638381	1	84.683605	1	0.952231
79.935934	1	84.772814	1	0.942943
80.695472	1	47.194654	2	1.709843
80.465559	1	46.787433	2	1.719811

80.515556	1	47.409209	2	1.698310
80.892150	1	47.294200	2	1.710403
80.425375	1	45.364132	2	1.772885
80.445940	1	25.767286	4	3.122018
79.625988	1	25.252511	4	3.153191
80.482277	1	25.728727	4	3.128110
80.222821	1	25.733798	4	3.117411
80.320066	1	25.735781	4	3.120949

Da tabela pode-se notar que a multiplicação sequencial, para qualquer ordem de matriz, sempre vence a execução concorrente com 1 thread. Isto é, a multiplicação sequencial tem sempre um tempo menor que a concorrência com 1 thread e o ganho  $\frac{T_{seq}}{T_{conc}}$ , nesse caso, portanto, é sempre menor que 1.

E também nota-se como a concorrência com mais threads (2 ou 4) acelera a aplicação. Um ganho da ordem 2 faz com que o tempo concorrente seja a metade do tempo sequencial e um ganho da ordem de 3 faz com que o tempo concorrente seja três vezes menor que o tempo sequencial, que são melhorias bem significativas.

E por fim, a partir das informações de *tabela.txt* gera-se o conjunto de dados, usados pelo *gnuplot* para plotar os gráficos.

## Gráficos

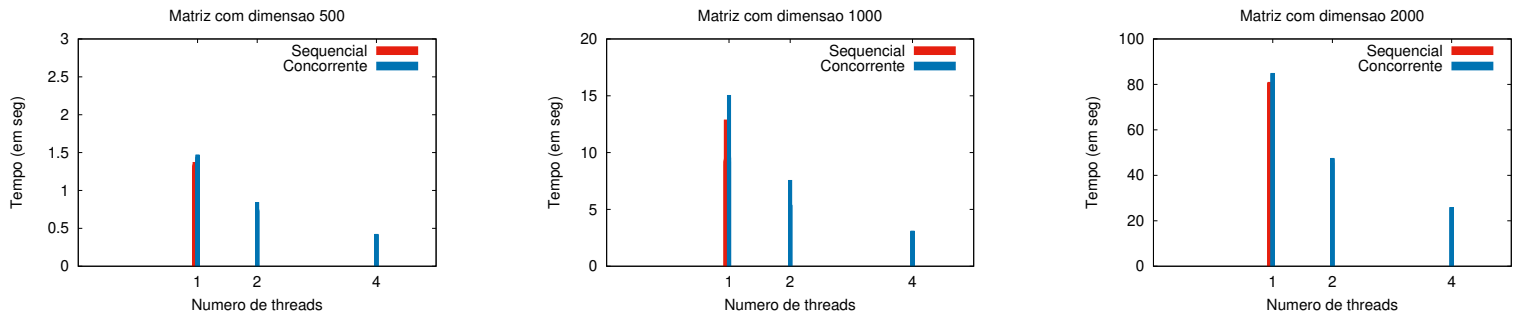


Figura 1: Tempo Sequencial e Concorrente das Matrizes